What do we know about software security evaluation? A preliminary study

Séverine Sentilles
School of Innovation, Design and Engineering, Mälardalen University
Västerås, Sweden
severine.sentilles@mdh.se

Efi Papatheocharous
ICT SICS
RISE Research Institutes of Sweden
Stockholm, Sweden
efi.papatheocharous@ri.se

Federico Ciccozzi
School of Innovation, Design and Engineering, Mälardalen University
Västerås, Sweden
federico.ciccozzi@mdh.se

***Abstract*—In software development, software quality is nowadays acknowledged to be as important as software functionality and there exists an extensive body-of-knowledge on the topic. Yet, software quality is still marginalized in practice: there is no consensus on what software quality exactly is, how it is achieved and evaluated. This work investigates the state-of-the-art of software quality by focusing on the description of evaluation methods for a subset of software qualities, namely those related to software security. The main finding of this paper is the lack of information regarding fundamental aspects that ought to be specified in an evaluation method description. This work follows up the authors' previous work on the Property Model Ontology by carrying out a systematic investigation of the state-of-the-art on evaluation methods for software security. Results show that only 25% of the papers studied provide enough information on the security evaluation methods they use in their validation processes, whereas the rest of the papers lack important information about various aspects of the methods (e.g., benchmarking and comparison to other properties, parameters, applicability criteria, assumptions and available implementations). This is a major hinder to their further use.**

***Index Terms*—Software security, software quality evaluation, systematic review, property model ontology.**

I. Introduction

Software quality measurement quantifies to what extent a software complies with or conforms to a specific set of desired requirements or specifications. Typically, these are classified as: (i) functional requirements, pertaining to what the software delivers, and (ii) non-functional requirements, reflecting how well it performs according to the specifications. While there exists a vast plethora of options for functional requirements, non-functional requirements have been studied extensively and classified through standards and models (i.e., ISO/IEC 9126 [1], ISO/IEC 25010 [2] and ISO/IEC 25000 [3]). They are often referred to as extra-functional properties, non-functional properties, quality properties, quality of service, product quality or simply metrics.

Despite the classification and terminology accompanying these properties, their evaluation and the extent of how well a software system performs under specific circumstances are both hard to quantify without substantial knowledge on the particular context, concurring measurements and measurement methods. In practice, quality assessment is still marginalized – there is no consensus on what software quality exactly is, how it is achieved and evaluated.

In our research we investigate to what extent properties and their evaluation methods are explicitly defined in the existing literature. We are interested in properties such as reliability, efficiency, security, and maintainability. In our previous work [4], we investigated which properties are most often mentioned in literature and how they are defined in a safety-critical context. We found that the most prominent properties of cost, performance and reliability/safety were used, albeit not always well-defined by the authors: divergent or non existent definitions of the properties were commonly observed.

The target of this work is to investigate evaluation methods related to software security, which is a critical extra-functional property directly and significantly impacting different development stages (i.e., requirements, design, implementation and testing). Security is defined by McGraw [5] as "engineering software so that it continues to function correctly under malicious attack". The high diversity of components and complexity of current systems makes it almost impossible to identify, assess and address all possible attacks and aspects of security vulnerabilities.

More specifically, in this paper we want to answer the following research question: *"What percentage of evaluation method descriptions contains pertinent information to facilitate its use?"*. For this, we identified a set of papers representing the state-of-the-art from the literature in the field and we assessed the degree of explicit information about evaluation methods. The set of papers was analyzed to identify definition and knowledge representation issues with respect to security, and answer the following three sub-questions:

- RQ1: What proportion of properties is explicitly defined?
- RQ2: What proportion of evaluation methods provide explicit key information to enable their use (e.g., formula, description)?
- RQ3: What proportion of other supporting elements of the evaluation methods is explicit (e.g., assumptions, available implementations)?

We used a systematic mapping methodology [6], started from 63 papers and selected 24 papers, which we then analyzed further. After thorough analysis, we excluded eight papers, resulting in a final set of 16 papers, which we used for synthesizing and reporting our results. The final set of papers is listed in Table V. The main contribution of this paper is represented by the identification of the following aspects:

- relation between evaluation methods and a generic property definition (i.e., if it exists explicitly and how it is described),
- which aspects of other supporting elements to improve understandability and applicability of the method (i.e, assumptions, applicability, output etc.) are explicitly mentioned, and
- the missing information regarding fundamental aspects which ought to be specified in a property and method description.

The remainder of the paper is structured as follows. In Section II we introduce the related work in the topic, while in Section III we describe our methodology. Section IV summarizes quantitative and qualitative interpretations of the extracted data, while Section V reports the main results. Section VI discusses the threats to validity and Section VII concludes the paper and delineates possible future directions.

II. Related work

An extensive body-of-knowledge already exists on software security and many studies have already investigated security assessment as well. For example, the U.S. National Institute of Information Standards and Technology (NIST) in [7] developed a performance measurement guide for information security and in particular how an organization through the use of metrics can identify the adequacy of controls, policies and procedures. The approach is mostly focused on the level of technical security controls in the organization rather than the technical security level of specific products.

In [8], a taxonomy for information security-oriented metrics is proposed in alignment to common business goals (e.g., cost-benefit analysis, business collaboration, risk analysis, information security management and security dependability and trust for ICT products, systems and services) covering both organizational and product levels.

Verendel [9] analyzed quantitatively information security from a set of articles published from 1981-2008 and concluded that theoretical methods are difficult to apply in practice and there is limited experimental repeatability. A recent literature survey [10] carried out on ontologies and taxonomies of security assessment identified among others a gap in works addressing research issues like knowledge reuse, automatic processes, increasing the assessment coverage, defining security standards and measuring security.

Morrison et al. [11] carried out a systematic mapping study on software security metrics to create a catalogue of metrics, their subject of measurement, validation methods and mappings based on the software development life cycle (SDLC). Based on the vast catalogue of metrics and definitions collected, a major problem is that they are extremely hard to be compared, as they typically measure different aspects of the property (something obvious from the emergent categories proposed in the paper). Thus, there is no agreement on their degree of assessment coverage or how to achieve their evaluation. Moreover, in their work, the generic property of security, including its definition, relation to a reference and detailed explanation on how to achieve assessment, is not addressed.

Therefore, we decided to investigate thoroughly the literature and quantify the quality of the description of security evaluation methods. Our approach differs from the work of Morrison et al. [11] in that we are not interested to collect metrics and their mapping to SDLC. Instead, we examine evidence on properties' explicit definitions, their evaluation methods and parameters (including their explanations on how they are used) for assessing security. We develop a mapping from an applicability perspective of the evaluations, which may be used by researchers and practitioners at different phases of security evaluations, i.e., at the low-level internal or at the external (often called operational) phase of software engineering.

III. Methodology

We base our work on the mapping study previously conducted by Morrison et al. in [11]. The reasoning is that the basis of the work is well aligned and applicable to answering our RQs. The details of the methodology are explained in the following.

A. Search and selection strategy

The data sources used in [11] are the online databases, conference proceedings and academic journals of ACM Digital Library, IEEE Xplore and Elsevier. The search terms include the keywords: "software", "security", "measure", "metric" and "validation". Considering another 31 synonyms, the following search string was used:

"`(security OR vulnerability) AND (metric OR measure OR measurement OR indicator OR attribute OR property)`",

in which the terms are successively replaced with the identified synonyms.

The selection criteria of Morrison et al. include the sets of inclusion and exclusion items listed below.

Inclusion criteria:

- Paper is primarily related to measuring software security in the software development process and/or its artifacts. For example software artifacts (e.g., source code files, binaries), software process (e.g., requirements phase, design, implementation, testing), and/or software process artifacts (e.g., design and functional specifications).
- Measurements and/or metrics are the main paper subject
- Refereed paper
- Paper published since 2000

Exclusion criteria:

- Related to sensors
- Related to identity, anonymity, privacy
- Related to forgery and/or biometrics
- Related to network security (or vehicles)

- Related to encryption
- Related to database security
- Related to imagery, audio, or video
- Specific to single programming languages

As explained above, search string, search strategy, inclusion and exclusion criteria defined by Morrison et al. are applicable to our work. Thus, we reuse the list of selected papers from their mapping study as the initial set for our study. However, as we wanted to investigate the applicability and quality of the security evaluation methods found, we extended the selection and data extraction methodology to isolate the subset of selected papers that is considered most relevant. In order to identify the subset, we performed the following actions:

- Define new exclusion criteria
- Design a new selection process
- Define a new data collection scheme
- Extract the data
- Intepret data and synthesize results

B. *New exclusion criteria*

We decided to exclude sources according to the following criteria:

- Full-text is not available
- Sources are Ph.D. dissertations or books
- Sources are not industrially validated or do not use well-known software, tool or platforms for their validations
- Sources are model predictions that do not assess any property

C. *New selection process*

The selection is performed on the list of 63 selected papers from [11] by three independent researchers by looking at the title, abstract and by going through a quick reading of the content. Papers independently selected by all researchers are included. Similarly, papers which are discarded by all researchers are excluded. For papers for which there is a disagreement, a discussion is held between the involved researchers to unanimously decide whether to include or exclude the papers. In one case (p18), an extended version of the paper was found and used instead (i.e. we used [12] instead of [13]). This selection process resulted in 24 included papers and 39 rejected. Furthermore, 8 papers were later excluded during the data extraction as they did not contain useful information or were not of sufficient quality. Hence, our selection set is based on the 16 papers listed in Table V.

D. *Data collection scheme*

As support for answering the RQs, we created a form based on excel spreadsheets. The form consists of the questions specified in Table I, with the list of possible values for each question. The questions are derived from our previous work on the Property Model Ontology (PMO), which formally specifies which concepts should be described for properties and their evaluation methods and how these concepts relate to one another. For details on the PMO, the reader is directed to [4].

E. *Data extraction and interpretation*

The data extraction is carried out by the researchers using independent spreadsheet columns (one for each paper). The data extraction for each paper is reviewed by another researcher (i.e. reviewer) and discussions are carried out until an agreement is reached. The data is then interpreted and analyzed by all researchers together, as explained in the next section.

IV. Data interpretation

A. *Quantitative analysis*

One interesting, albeit not very surprising fact, is that most of the descriptions contained in the papers explicitly state what is supposed to be calculated (i.e., the property and the output) and how (i.e., through the overall approach description, the parameters involved in the approach, and to some extent the advantages and disadvantages). As visible in Table II, which shows the results of the quantitative analysis, the property is explicitly referred to in 87,5% of the papers, the output in 75%, the approach description in 87,5%, the parameters in 81%, the advantages in 62% and the disadvantages in 69%.

No method explicitly specifies the unit to be used for the output. This might be explained by the fact that software security is a rather recent research field and there is currently no widely acknowledged metric and corresponding units to be used. As a result, most methods typically fall back on standard data types for which units are less relevant (e.g., percentage, probability, item counts).

Applicability criteria are rarely explicitly mentioned, only in 19% of the papers. When they are mentioned, it is not obvious whether the set of provided criteria is complete or at least sufficient to ensure the applicability of the evaluation method.

Only half of the papers clearly specify the assumptions and hypotheses that are assumed to hold when using the method. This is a setback as these aspects are important for correctly applying a method.

Drivers are rarely explicitly mentioned; only in 12.5% of the papers. Drivers can be important as they can implicitly affect the output of a method.

Despite being often mentioned in the papers as part of the solution being implemented, in practice, only a few implementations or programs are available to directly support the method evaluation. We could find mentions of available tool support in only 2 out of the 16 papers (i.e., WEKA toolkit and Fortify Source Code Analyzer (SCA) version 5.10.0.0102 in p43 and p47). A few additional papers (4) refer to available tool support but no explicit link or reference is provided.

Out of the 16 analyzed papers, only 1 (i.e., p60) performed some kind of benchmarking or comparison to similar properties or evaluation methods. This corresponds to only 6% of all the papers.

To a large extent, the information provided in the papers is insufficient to directly apply the method, especially by non-experts (in 12 papers, 75%).

TABLE I: Data collection form

	Data to collect	**Possible Values**
q1	PaperId	Px, with x a number
q2	Method name	Free text
q3	Does the evaluation method provide an explicit reference to a property or a metric	Yes | no
q4	If yes on q3, how?	Name + def. + ref. | name + def. | name + ref. | other
q5	If other on q4, how?	Free text
q6	If yes on q3, which one? Give the example	Free text
q7	If yes on q3, does the reference match what is needed for the evaluation method?	Yes | no | I don't know
q8	Does the evaluation method explicitly state the output of the method	Yes | no
q9	If yes on q8, how?	With data format and unit | With data only | other
q10	If other on q9, how?	Free text
q11	If yes on q8, which one? Give the example	Free text
q14	Does the evaluation method explicitly state applicability criteria?	Yes | no
q15	If yes, which ones?	Free text
q16	Does the evaluation method explicitly state how the property is evaluated?	Yes | no | I don't know
q17	If yes on q16, how?	Free text
q18	Does the evaluation method explicitly explain the parameters involved in the evaluation?	Yes | no
q19	If yes on q19, which parameters (with their explanations)?	Free text
q25	Does the evaluation method explicitly describe additional drivers that might affect the evaluation?	Yes | no
q26	If yes on q25, which ones?	Free text
q27	Does the evaluation method explicitly state the assumptions, hypotheses which the method is based on?	Yes | no
q28	If yes on q27, which ones?	Free text
q29	Does the evaluation method explicitly mentioned advantages for the method?	Yes | no
q30	If yes on q30, which ones?	Free text
q31	Does the evaluation method explicitly mentioned disadvantages for the method	Yes | no
q32	If yes on q31, which ones?	Free text
q33	Does the evaluation method explicitly reference an implementation or a program that can be used?	Yes | no
q34	If yes on q33, which ones?	Free text
q35	If yes on q33, is the implementation or program accessible	Yes | no
q36	Is there any comparison to other properties (e.g., benchmark, validation)	Yes | no
q37	If yes on q36, how?	Free text
q38	Additional comments	Free text
q39	Is the information provided in the paper sufficient to understand the method and directly use it?	Yes | no
q40	Extractor's name	Séverine | Efi | Federico
q41	Reviewer's name	Séverine | Efi | Federico

B. Qualitative analysis

From analyzing the answers to the questions in Table II, papers can be categorized into three groups based on their main purpose. The first group focuses on defining a new property or metric to assess some specific software security aspects (p1, p18, p27, p47, p63). The papers belonging to the second group (p15, p25, p37, p51, p60) base their work on already defined properties. Their main objective is to either find ways to combine existing metrics to evaluate a given security aspect, define a new evaluation method for existing properties or validate previously specified methods in applying them on a specific system. The last group aims at finding correlation between already defined properties or performing predictions to evaluate the accuracy of previously defined models (p38, p43, p53, p55, p59). There are papers that do not belong to these groups as they are not explicitly referring to any property, method or metric (p22, p50). This is shown in the answers to questions on whether the paper explicitly refers to a property and, if yes, how (i.e., q3 and q4). Papers belonging to the first group use "definition without reference", papers from the second group use "definition plus reference" and papers from the last group use "other".

From the data extracted, 34 definitions and references on security are collected from the papers, as listed in Table IV. We categorized the collected properties in two different levels, depending on their level of specificity (less coarse-grained). A few papers shared some definitions (e.g., p25, p59, p63) whereas most papers defined their own security aspect to evaluate. The security aspect expressed (even if in some cases is explained with many details) is only able to capture some facets of the property, thus it is hard to know if the property definitions together with their evaluations are enough to asses security in a rational way.

Related to the descriptions of the properties and evaluations extracted, we captured the level of applicability between them, as they were described in the papers: most commonly the deployment and operation phase (post-release), the implementation (development or code level, including testing) and the design phase as shown in Table III. The level at which security evaluation is carried out varies among the papers. 7 of them (44%) apply the evaluation method at the implementation (code-level) phase, 1 paper at the testing phase, 6 papers (37,5%) at the operational phase, 1 paper during maintenance and 4 papers (25%) at the design level.

TABLE II: Quantitative results

Question	Count of 'Yes' (%)	Count of 'No' (%)
q3: Explicit property	14 (87,5%)	2 (12,5%)
q8: Explicit method output	12 (75%)	4 (25%)
q14: Explicit applicability criteria	3 (19%)	13 (81%)
q16: Explicit description	14 (87,5%)	2 (12,5%)
q18: Explicit method parameters	13 (81%)	3 (19%)
q25: Explicit drivers	2 (12,5%)	14 (87,5%)
q27: Explicit assumptions	8 (50%)	8 (50%)
q29: Explicit advantages	10 (62,5%)	6 (37,5%)
q31: Explicit disadvantages	11 (69%)	5 (31%)
q33: Explicit tool reference	6 (37,5%)	10 (62,5%)
q35: Tool accessibility	2 (12,5%)	4 (25%)
q36: Benchmark	1 (6%)	15 (94%)
q39: Information is sufficient	4 (25%)	12 (75%)

TABLE III: Security level applicability

pID	Applicability level	Analysis level*
p1	system architecture - design - implementation	internal
p15	implementation	internal
p18	operational	external
p22	operational - system architecture - design	int. & ext.
p25	operational	int. & ext.
p27	operational	external
p37	implementation	internal
p38	design	internal
p43	design - implementation	internal
p47	operational	external
p50	operational	external
p51	implementation	internal
p53	maintenance	external
p59	testing	external
p60	detailed design - implementation	internal
p63	implementation	internal

*Based on the definitions from [1] on internal and external metrics for product quality.

As mentioned above, no units are explicitly specified. However, the output of the evaluation methods falls back onto implicit scales of measurements: nominal (p60), ordinal (p1, p18, p37, p47, p50, p60, p63), interval (p53) and ratio (p18, p25, p43, p60).

Regarding advantages of the methods, the most commonly reported are: reliability of the method (p1, p15, p25), simplicity of the method (p1, p18, p25), accuracy of the method (p18, p43) and objectivity of the results (p47, p60). The disadvantages mostly refer to the accuracy of the results being dependent on the quality of the available data (p18, p25, p37, p43) and subjectivity involved in the method (p22, p37, p63).

V. Results and discussions

As an answer to our initial research question "What percentage of evaluation methods description contains pertinent information to facilitate its use?", only 25% of the papers that we investigated were judged to provide enough information to enable a direct application of the method. Overall, the papers were good at explicitly stating the property under study, describing how the property is evaluated and which parameters are involved. On the other hand, in several cases the papers lacked information regarding the unit representing the value of the property, the applicability criteria for the method as well as its assumptions, possible advantages and disadvantages, eventual openly available tool support, and comparison to other properties.

Security being a relatively new area in software engineering can be a major contributing factor to these results. For example, advantages, disadvantages, and applicability criteria are difficult to identify initially. They require time and perspective on the topic as well as the methods having been used over time in a wide range of applications and scenarios. Given that many of the papers which were included in our set focused on defining new properties for security, it is not so surprising that so few papers mentioned those aspects. Furthermore, following the same reasoning, even if those criteria were explicitly stated in the papers, it is not certain that the proposed lists of advantages, disadvantages and applicability is exhaustive, or at least sufficient to guarantee the proper usage of the evaluation method that they introduce.

When comparing to the results in [11], we found fewer metrics and definitions. This is due to the fact that our purpose was to identify the main property (or properties) of the paper and their corresponding evaluation methods. Some of the metrics identified in [11] are classified in our results as parameters. Others have been ignored as they were used for comparison purposes and therefore not as relevant for our work. However, the conclusions by Morrison et al. still hold and are further supported by our results. Security properties are not mature, most of them have been proposed and evaluated solely by their authors, and few comparisons between already defined properties exist. "Despite the abundance of metrics found in the literature, those available give us an incomplete, disjointed, hazy view of software security."

These results are to be put in perspective, since none of the authors of this paper is an expert in software security and due to the small number of papers that were studied in this work. However, we are confident that the observations resulting from this study are good indications of the issues occurring with property and evaluation methods descriptions in software security.

A bi-product of our analysis is the following interesting aspect. Especially for the works assessing software vulnerability, evaluation methods exploit well-established prediction models that leverage a discernible set of software metrics (related to the code itself, developers activity, versioning data, etc.). A (what seems to be) common step to the definition of security-related evaluation methods, especially when dealing with vulnerability, is the comparison of existing prediction models, or the comparison of a newly defined prediction model with a set of existing ones, in order to identify the "best" model

TABLE IV: Collection of properties or coarse-grained metrics explicitly defined in the papers[1]

pID	Generic/coarse-grained definition	Less generic definition
p1	Security to mean control over data confidentiality [...] and data integrity [...][1]	Total Security Index (TSI) as the sum of the security design principle metrics
p15	Vulnerability as a weakness in a software system that allows an attacker to use the system for a malicious purpose	
p18	End-user software vulnerability exposure as a combination of lifespans and vulnerability announcement rates	Median Active Vulnerabilities (MAV): the median number of software vulnerabilities which are known to the vendor of a particular piece of software but for which no patch has been publicly released by the vendor Vulnerability Free Days (VFD): captures the probability that a given day has exactly zero active vulnerabilities
p25	Vulnerability (defined in [14])	
p27	Operational security as representation of as accurately as possible the security of the system in operation, i.e., its ability to resist possible attacks or, equivalently, the difficulty for an attacker to exploit the vulnerabilities present in the system and defeat the security objectives	Mean Effort To security Failure (METF): Mean effort for a potential attacker to reach the specified target
p37	Security through an attack surface metric	Attack surface metric a measure of a systems attack surface along three dimensions by estimating the total contribution of the methods, the total contribution of the channels, and the total contribution of the data items to the systems attack surface [...][1]
p38	Security as through the number of violations of the least privilege principle and surface metrics Maintainability as coupling between components and components instability	LP principle metric (defined in [15]) Attack surface metric (defined in [16]) Coupling between components (CBM) (defined in [17]) Components instability (CI) (defined in [18])
p43	Vulnerability through dependency graphs (*)	
p47	Vulnerability through static analysis (*)	Static-analysis vulnerability density (SAVD): number of vulnerabilities a static-analysis tool finds per thousand LOC Density of post-release reported vulnerability (NVD) (*) Application vulnerability rank (SAVI) (*)
p50	Vulnerability as flaws or weakness in a systems design, implementation, or operation and management that could be exploited to violate the systems security policy. Any flaw or weakness in an information system could be exploited to gain unauthorized access to, damage or compromise the information system.	
p51	Application security (defined in [5]) Software security (defined in [5]) Security at program level (defined in [19])	Stall Ratio (SR): a measure of how much a programs progress is impeded by frivolous activities. Coupling Corruption Propagation (CCP): Number of child methods invoked with the parameter(s) based on the parameter(s) of the original invocation Critical Element Ratio (CER) (*)
p53	Security through vulnerability density (*)	Static analysis vulnerability density (SAVD) (see p47) Security Resources Indicator (SRI): the sum of four indicator items, ranging from 0 to 4. The items are: the documentation of the security implications of configuring and installing the application, a dedicated e-mail alias to report security problems, a list or database of security vulnerabilities specific to the application, and the documentation of secure development practices, such as coding standards or techniques to avoid common secure programming errors.
p59	Vulnerability (defined in [14])	
p60	Software vulnerability (defined in [19])	Structural severity: uses software attributes to evaluate the risk of an attacker reaching a vulnerability location from attack surface entry points [...][1] Attack Surface Entry Points (defined in [20]) Reachability Analysis (*)
p63	Software vulnerability (defined in [14])	Vulnerability-Contributing Commits (VCCs): original coding mistakes or commits in the version control repository that contributed to the introduction of a post-release vulnerability

[1] The definitions have been shorten to comply with the page limitation. Readers are referred to the original paper for the complete definition.
(*) No explicit definition is found.

to use as basis for evaluation purposes. Another interesting aspect is represented by the fact that, in several papers, authors exercise their reasoning and methods on well-known codebases (e.g., Windows Vista, Mozilla); this shows an interesting strong inclination towards assessing the applicability of research (theoretical) results to practical cases, which is too seldom seen in other research branches within software engineering.

VI. Threats to validity

Construct validity relates to what extent the phenomenon under study represents what the researchers wanted to investigate and what is specified by the research questions. We explicitly defined the context of the work and discussed related terms and concepts. Also, the research questions were formulated based on these clarified notions and the research followed a methodological procedure known as systematic mapping.

The selection of papers was based on the work of Morrison et al. [11], thus we inherit the work's limitations. Therefore, papers not listed in the sources used, i.e., ACM, IEEE and Elsevier, have been missed. In addition, we excluded Ph.D. dissertations and books since we limited our selection to peer-reviewed conference and scholar journal publications only.

The way the involved researchers individually interpreted the methods described in the papers reflects their own biases and views. However, we worked hard to reduce the bias by discussing the content of the papers in pairs to dissolve uncertainties. In several cases, where the decision to include or not a paper, a third research was involved to reach a consensus.

Morrison at al. [11] excluded sources dealing with networks, sensors, database and vehicle security and this limited the set of papers analyzed in our study. This opens however up for opportunities of further research targeting these particular types of systems.

External validity is about to what extent the findings are generalizable. Due to the focus on the security aspect and the small selection of papers, one should avoid generalizing the results over other properties such as reliability and safety. However, despite the limitations of our study (i.e., low number of papers, one method per paper, no snowballing to find complementary information), our conclusions are representative of the current issues in software security. Furthermore, other works in the state-of-the-art on software quality share similar conclusions as Morrison et. al. in [11], which points towards the applicability of our results to other properties.

VII. Conclusions

Despite the low number of papers this work is based on, we believe it is, to some extent, representative of the current issues in the state-of-the-art on software quality in general and security in particular: a number of useful properties and methods to evaluate them do exist today. However, it is difficult to know about them and understand whether they are applicable in a given context and if they are, how to use them. This can be attributed to the lack of information in the descriptions of their evaluation methods. This hampers the activities towards better quality assurance in software engineering and it limits at the same time the application of these activities or newly specified methods in industrial settings. For example, knowing when to apply a method (e.g., applicability at design time, at run-time, etc.) restricts which methods can be used in a given context. Knowing the advantages and disadvantages allows to trade-off available methods and limits selection bias. Older, more established or traditional, software quality fields provide more reference properties and methods to systematically compare to; however the level of detail and quality of given information is still relatively low.

As future work, we plan to expand the selection of papers to include those from the references in the analyzed papers so to investigate if our conclusions still hold. Similarly, we will explore the quality of assessments of other quality properties in literature such as reliability, safety and maintainability. The results of these studies will be included in PROMOpedia [21], an online encyclopedia of software properties and their evaluation methods. Lastly, we plan to propose an improved and validated ontology to express several critical and time sensitive properties towards a more systematic (in terms of consistent) and formal (in terms of codified) way and approach a better trade-off support between the properties.

Part of the work is also supported by the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway.

Acknowledgments

The work is supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden.

References

[1] ISO/IEC, *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001.

[2] ——, *Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*, 2011.

[3] ——, "ISO/IEC 25000 software and system engineering–software product quality requirements and evaluation (SQuaRE)–guide to SQuaRE," *International Organization for Standardization*, 2005.

[4] S. Sentilles, E. Papatheocharous, F. Ciccozzi, and K. Petersen, "A property model ontology," in *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*. IEEE, 2016, pp. 165–172.

[5] G. McGraw, *Software security: building security in*. Addison-Wesley Professional, 2006, vol. 1.

[6] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering." in *EASE*, vol. 8, 2008, pp. 68–77.

[7] E. Chew, M. M. Swanson, K. M. Stine, N. Bartol, A. Brown, and W. Robinson, "Performance measurement guide for information security," Tech. Rep., 2008.

[8] R. Savola, "Towards a security metrics taxonomy for the information and communication technology industry," in *Software Engineering Advances, 2007. ICSEA 2007. International Conference on*. IEEE, 2007, pp. 60–60.

[9] V. Verendel, "Quantified security is a weak hypothesis: a critical survey of results and assumptions," in *Proceedings of the 2009 workshop on New security paradigms workshop*. ACM, 2009, pp. 37–50.

[10] F. d. F. Rosa, R. Bonacin, and M. Jino, "The security assessment domain: A survey of taxonomies and ontologies," *arXiv preprint arXiv:1706.09772*, 2017.

[11] P. Morrison, D. Moye, and L. A. Williams, "Mapping the field of software security metrics," North Carolina State University. Dept. of Computer Science, Tech. Rep., 2014.

[12] J. L. Wright, M. McQueen, and L. Wellman, "Analyses of two end-user software vulnerability exposure metrics (extended version)," *Information Security Technical Report*, vol. 17, no. 4, pp. 173–184, 2013.

[13] ——, "Analyses of two end-user software vulnerability exposure metrics," in *2012 Seventh International Conference on Availability, Reliability and Security*. IEEE, 2012, pp. 1–10.

[14] I. V. Krsul, *Software vulnerability analysis*. Purdue University West Lafayette, IN, 1998.

[15] K. Buyens, B. De Win, and W. Joosen, "Identifying and resolving least privilege violations in software architectures," in *Availability, Reliability and Security, 2009. ARES'09. International Conference on*. IEEE, 2009, pp. 232–239.

[16] P. K. Manadhata, D. K. Kaynar, and J. M. Wing, "A formal model for a system's attack surface," CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, Tech. Rep., 2007.

[17] M. Lindvall, R. T. Tvedt, and P. Costa, "An empirically-based process for software architecture evaluation," *Empirical Software Engineering*, vol. 8, no. 1, pp. 83–108, 2003.

[18] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[19] C. P. Pfleeger and S. L. Pfleeger, *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

[20] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, no. 3, pp. 371–386, 2010.

[21] S. Sentilles, F. Ciccozzi, and E. Papatheocharous, "PROMOpedia: a web-content management-based encyclopedia of software property models," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. ACM, 2018, pp. 45–48.

TABLE V: List of selected papers with their ID

pID	Paper reference
p1	B. Alshammari, C. Fidge, and D. Corney, “A hierarchical security assessment model for object-oriented programs,” in Quality Software(QSIC), 2011 11th International Conference on. IEEE, 2011, pp. 218227.
p15	Y. Shin and L. Williams, “An empirical model to predict security vulnerabilities using code complexity metrics,” inProceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. ACM, 2008, pp. 315317.
p18	J. L. Wright, M. McQueen, and L. Wellman, “Analyses of two end-user software vulnerability exposure metrics (extended version),” Information Security Technical Report, vol. 17, no. 4, pp. 173184, 2013
p22	M. Almorsy, J. Grundy, and A. S. Ibrahim, “Automated software architecture security risk analysis using formalized signatures,” in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013, pp. 662671.
p25	Shin, A. Meneely, L. Williams, and J. A. Osborne, “Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities,” IEEE Transactions on Software Engineering,vol. 37, no. 6, pp. 772787, 2011.
p27	R. Ortalo, Y. Deswarte, and M. Kaaniche, “Experimenting with quantitative evaluation tools for monitoring operational security,” IEEE Transactions on Software Engineering, no. 5, pp. 633650, 1999.
p37	P. Manadhata, J. Wing, M. Flynn, and M. McQueen, “Measuring the attack surfaces of two FTP daemons,” in Proceedings of the 2nd ACMworkshop on Quality of protection. ACM, 2006, pp. 310.
p38	K. Buyens, R. Scandariato, and W. Joosen, “Measuring the interplay of security principles in software architectures,” in Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society, 2009, pp. 554563.
p43	V. H. Nguyen and L. M. S. Tran, “Predicting vulnerable software components with dependency graphs,” in Proceedings of the 6th International Workshop on Security Measurements and Metrics. ACM, 2010, p. 3.
p47	J. Walden and M. Doyle, “Savi: Static-analysis vulnerability indicator,” IEEE Security & Privacy, no. 1, 2012.
p50	J. A. Wang, H. Wang, M. Guo, and M. Xia, “Security metrics for software systems,” in Proceedings of the 47th Annual Southeast Regional Conference. ACM, 2009, p. 47.
p51	I. Chowdhury, B. Chan, and M. Zulkernine, “Security metrics for source code structures,” in Proceedings of the fourth international workshop on Software engineering for secure systems. ACM, 2008, pp. 5764.
p53	J. Walden, M. Doyle, G. A. Welch, and M. Whelan, “Security of open source web applications,” in Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on.IEEE, 2009, pp. 545553
p59	M. Gegick, P. Rotella, and L. Williams, “Toward non-security failures as a predictor of security faults and failures,” in International Symposium on Engineering Secure Software and Systems. Springer, 2009, pp. 135149.
p60	A. A. Younis, Y. K. Malaiya, and I. Ray, “Using attack surface entry points and reachability analysis to assess the risk of software vulnerability exploitability,” in High-Assurance Systems Engineering (HASE),2014 IEEE 15th International Symposium on. IEEE, 2014, pp. 18
p63	A. Meneely, H. Srinivasan, A. Musa, A. R. Tejeda, M. Mokary, and B. Spates, “When a patch goes bad: Exploring the properties of vulnerability-contributing commits,” in Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on. IEEE,2013, pp. 6574.